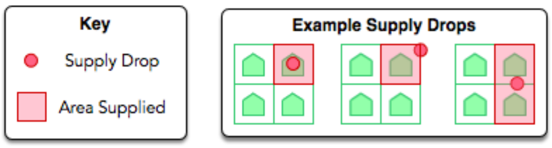
Army Game

Luke is daydreaming in Math class. He has a sheet of graph paper with n rows and m columns, and he imagines that there is an army base in each cell for a total of $n \cdot m$ bases. He wants to drop supplies at strategic points on the sheet, marking each drop point with a red dot. If a base contains at least one package inside or on top of its border fence, then it's considered to be supplied. For example:



Given n and m, what's the minimum number of packages that Luke must drop to supply all of his bases?

Example

$n = 2$
$m = 3$

Packages can be dropped at the corner between cells (0, 0), (0, 1), (1, 0) and (1, 1) to supply 4 bases. Another package can be dropped at a border between (0, 2) and (1, 2). This supplies all bases using 2 packages.

Function Description

Complete the *gameWithCells* function in the editor below.

gameWithCells has the following parameters:

- *int n:* the number of rows in the game
- *int m:* the number of columns in the game

Returns

- *int:* the minimum number of packages required

Input Format

Two space-separated integers describing the respective values of n and m.

Constraints

$0 < n, m \leq 1000$

Sample Input 0

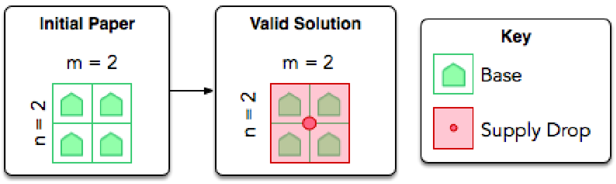
```
2 2
```

Sample Output 0

```
1
```

Explanation 0

Luke has four bases in a 2×2 grid. If he drops a single package where the walls of all four bases intersect, then those four cells can access the package:



Because he managed to supply all four bases with a single supply drop, we print 1 as our answer.

Contest ends in 5 days

Submissions: 17
Max Score: 20
Difficulty: Easy

Rate This Challenge:
☆☆☆☆☆

More

Pypy 3

```
#!/bin/python3

import math
import os
import random
import re
import sys

#
# Complete the 'gameWithCells' function below.
#
# The function is expected to return an INTEGER.
# The function accepts following parameters:
#  1. INTEGER n
#  2. INTEGER m
#

def gameWithCells(n, m):
    # Write your code here
    #i think need find a way to break into parts find biggets squre in the grid
    # nxm grid where m < n , we must find out how many dots for mxm then how many for an n-m, m
    # i think recursive? cuz i need find out how to do a n-m one too
    # first neeed find which is bigger


def square(m):
    out = m / 2
    wwwwww


if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')

    first_multiple_input = input().rstrip().split()

    n = int(first_multiple_input[0])

    m = int(first_multiple_input[1])

    result = gameWithCells(n, m)

    fptr.write(str(result) + '\n')

    fptr.close()
```

Line: 28 Col: 11

Upload Code as File ☐ Test against custom input Run Code Submit Code

Compile time error

Compile Message

```
Sorry: IndentationError: expected an indented block after function definition on line 18 (Solution.py, line 24)
```

Exit Status

```
1
```

Interview Prep | Blog | Scoring | Environment | FAQ | About Us | Support | Careers | Terms Of Service | Privacy Policy |